SQL
et
Bases de données

Cours 9

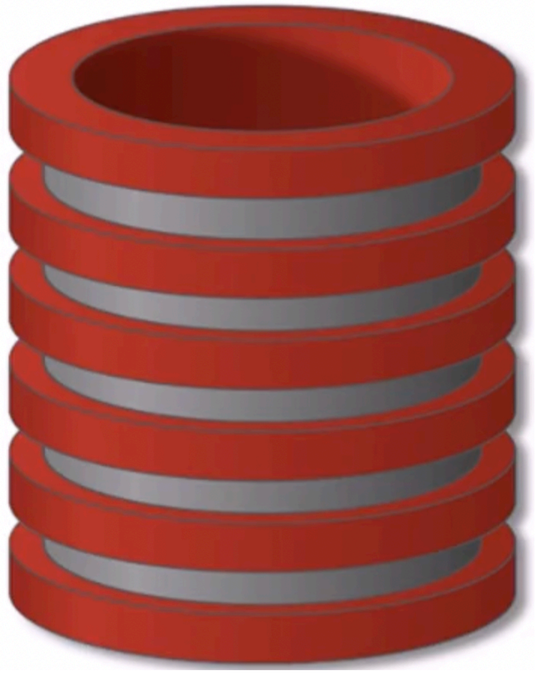
Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/lp-sql

Plan

- exercices (suite)
- calcul des relations
- dépendances et choix des relations



- deux bons tutoriels

  http://www.w3schools.com/sql/default.asp

  http://www.programiz.com/sql

client

cID	nom	actif	ville
1	Tom	23000	Bordeaux
2	Jean-Jacques	38000	Paris
3	Martin	51000	Nice
4	Kiki	54000	Pekin
5	Iteki	84000	Tokyo
6	Bob	6100	Nice
7	Albert	12000	Bordeaux
8	Manu	8150	Paris
9	Valou	10300	Bordeaux
10	Joe	32500	Nice
11	Helmut	8150	Paris
12	Martine	11200	Bordeaux
13	Marina	9150	Nice
14	Masha	10290	Nice
15	Julia	32000	Paris
17	Bob	38000	Nice

produit

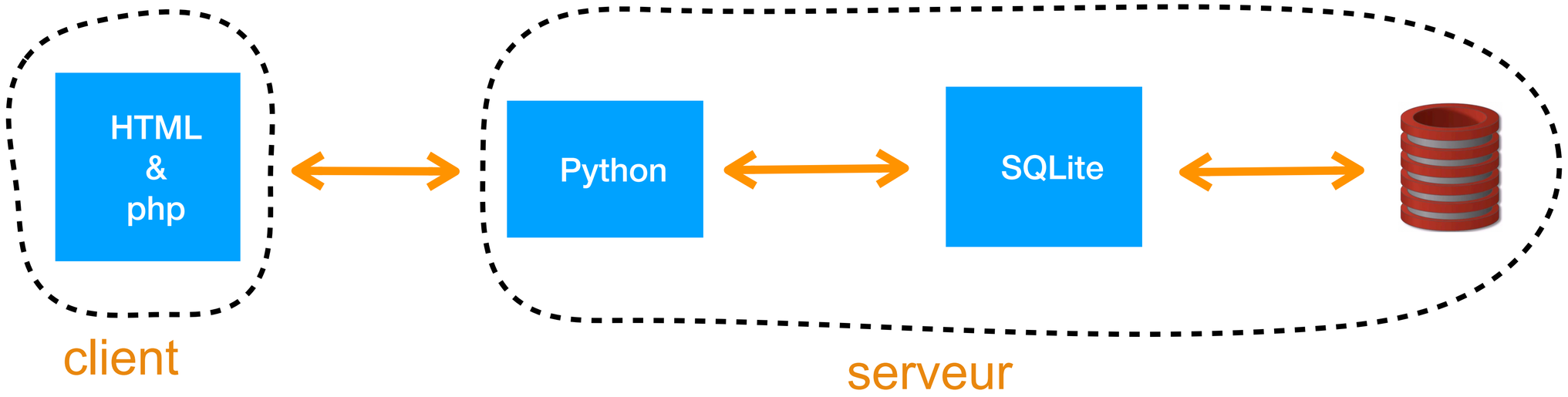
pID	pNom	pCat	pVille	prix
1	clio	auto	Paris	13000
2	audi	auto	Paris	45000
3	tesla	auto	Pekin	70000
4	tesla	auto	Nice	40000
5	yamaha	moto	Tokyo	8000
6	kawasaki	moto	Tokyo	8000
7	megamo	velo	Paris	3240
8	shimano	velo	Paris	1900
9	btwin	velo	Nice	990
10	triban	velo	Nice	690
11	peugeot	velo	Paris	750
12	bertin	eVelo	Paris	1190
13	trek	eVelo	Bordeaux	1390
14	trek	eVelo	Paris	1350

devis

cID	pNom	dCat	commande
2	peugeot	velo	0
7	NULL	velo	0
6	trek	eVelo	0
8	NULL	auto	0
8	NULL	velo	0
8	NULL	eVelo	0
9	honda	auto	0
11	NULL	auto	0
4	honda	moto	0
5	NULL	moto	0
8	triban	velo	0
13	NULL	velo	0
11	yaris	auto	0
12	NULL	velo	0
1	NULL	eVelo	0

Interface HTML — SQLite

- avec PHP, on peut appeler un serveur Python
- le serveur Python peut s'interfacer avec SQLite

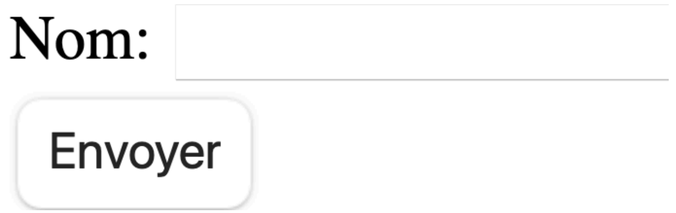



```
for row in cur.execute('select * from rlv order by opDate'):
        print(row)
```

Interface HTML — Python — SQLite

- un autre interface avec un formulaire texte

```
<html><body>

<form method="get" action=« http://localhost/cgi-bin/py1b.cgi" target="_blank">

Red<input type="checkbox" name="color" value="red">
Green<input type="checkbox" name="color" value="green">
<input type="submit" value="Submit">
</form>

<br><br>

<form action="http://localhost/cgi-bin/py2.cgi" method="get">
Nom: <input type="text" name="nom"><br>
<input type="submit" value="Envoyer">
</form>


</body></html>
```



Interface HTML — Python — SQLite

- le serveur `py1b-cgi` utilise le formulaire pour la requête SQL

```
#!/opt/local/bin/python
print ("Content-type: text/html;charset=utf-8\n\n")

print ("<html>")
print ("<head>")
print ('<link rel="stylesheet" href="http://jeanjacqueslevy.net/w3.css">')
print ("</head>")
print ("<body>")
# ---------------- début du traitement fichier CGI
import cgi, cgitb
cgitb.enable()
form = cgi.FieldStorage()

print ('<h2> Requête envoyée: </h2>')

for key in form.keys() :
    color = "&".join (form.getlist(key))

print ('<div class="w3-container w3-round-large w3-margin w3-'
       + color + '">')
for key in form.keys() :
    r = key + ": " + "&".join (form.getlist(key))
    print (r)
print ("</div>")
print ("<br>")

# ---------------- fin de l'en-tête
```

```
# ---------------- fin de l'en-tête
import sqlite3

con = sqlite3.connect("minisample.db")
cur = con.cursor()

print ('<h2> Table client: </h2>')

print ('<div class="w3-container w3-round-large w3-margin w3-khaki ">')
for row in cur.execute ("select * from client"):
    print(row)
    print("<br>")

con.commit()
print ("</div>")
print ("</body>\n</html>")
```

Calcul des relations

- une algèbre d’opérations sur des relations
- opérations sur les ensembles: *union*, *intersection*, *différence*
- opérations pour supprimer des parties d’une relation: *sélection* et *projection*
- opérations pour combiner 2 relations: *produit cartésien* et *jointure*
- opérations de *renommage* qui ne change pas la relation
- chaque opération produit une (nouvelle) relation

Calcul des relations

- une algèbre d'opérations sur des relations *R, S, T*

- opérations sur les ensembles: *union*, *intersection*, *différence* $R \cup S$, $R \cap S$, $R - S$

  [les relations *R, S* doivent avoir les mêmes attributs]

- opérations pour supprimer des parties d'une relation: *sélection* et *projection*

  projection:

  si $A_1, A_2, \ldots A_n$ sont les attributs de R, on peut projeter sur un sous-ensemble de ces attributs,

  on écrit par exemple $\pi_{A_2, A_5, A_{10}}(R)$

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

R

A	D
1	4
1	5
6	10
9	10

$\pi_{A,D}(R)$

- SQL

```
select A, D from R
```

Calcul des relations

- une algèbre d'opérations sur des relations *R, S, T*

- opérations sur les ensembles: *union*, *intersection*, *différence* $R \cup S$, $R \cap S$, $R - S$

  [les relations *R, S* doivent avoir les mêmes attributs]

- opérations pour supprimer des parties d'une relation: *sélection* et *projection*

  sélection:

  on sélectionne les n-uplets satisfiant C dans la relation R et

  on écrit $\sigma_C(R)$

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

R

A	B	C	D
1	2	3	4
1	2	3	5

$\sigma_{2A<D}(R)$

- SQL

```
select * from R where C
```

Calcul des relations

- une algèbre d'opérations sur des relations *R, S, T*
- opérations pour combiner 2 relations: *produit cartésien* et *jointure*

  produit cartésien:

  former tous les n-uplets avec un n-uplet de R et un n-uplet de S

  on écrit $R \times S$ pour cette nouvelle relation

A	B
1	2
3	4

(a) Relation R

B	C	D
2	5	6
4	7	8
9	10	11

(b) Relation S

A	$R.B$	$S.B$	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

(c) Result $R \times S$

- SQL

```
select * from R, S
```

Calcul des relations

- une algèbre d'opérations sur des relations *R, S, T*
- opérations pour combiner 2 relations: *produit cartésien* et *jointure*

jointure naturelle

former tous les n-uplets avec un n-uplet de R et un n-uplet de S qui s'accordent sur tous leurs attributs communs

on écrit $R \bowtie S$ pour cette nouvelle relation

A	B	C
1	2	3
6	7	8
9	7	8

(a) Relation U

B	C	D
2	3	4
2	3	5
7	8	10

(b) Relation V

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

(c) Result $U \bowtie V$

- SQL

```
select * from R join S ;
```

Calcul des relations

- une algèbre d'opérations sur des relations *R, S, T*
- opérations pour combiner 2 relations: *produit cartésien* et *jointure*

jointure conditionnelle (*theta join*)

former tous les n-uplets avec un n-uplet de R et un n-uplet de S qui s'accordent sur la condition C

on écrit $R \bowtie_C S$ pour cette nouvelle relation

A	B	C
1	2	3
6	7	8
9	7	8

(a) Relation U

B	C	D
2	3	4
2	3	5
7	8	10

(b) Relation V

A	$U.B$	$U.C$	$V.B$	$V.C$	D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

Figure 2.17: Result of $U \bowtie_{A<D} V$

- SQL

```
select * from R, S where C
```

Calcul des relations

- une algèbre d'opérations sur des relations *R, S, T* qui ont chacune des attributs
- opérations de **renommage** qui ne change pas la relation

pour renommer la relation R en S et les attributs de R en $A_1, A_2, \ldots A_n$

on écrit $\rho_{S(A_1,A_2,\ldots A_n)}(R)$

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

R

A'	B'	C'	D'
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

$S = \rho_{S(A',B',C',D')}(R)$

- SQL

```
select A as A', B as B', C as C', D as D' from R as S
```

Calcul des relations

- une algèbre d'opérations sur des relations *R, S, T* qui ont chacune des attributs $A_1, A_2, \ldots A_n$
- opérations qui produisent une nouvelle relation

$$R \cup S \qquad R \cap S \qquad R - S$$
$$\pi_{A_2,A_5,A_{10}}(R) \qquad R \times S \qquad R \bowtie_C S$$
$$\sigma_C(R) \qquad R \bowtie S \qquad \rho_{S(A_1,A_2,\ldots A_n)}(R)$$

SQL

- opérations qui produisent une nouvelle relation

$$\sigma_{C_1.\text{actif} = C_2.\text{actif} \wedge C_1.\text{cID} < C_2.\text{cID}} \quad (\rho_{C_1}(\text{client}) \times \rho_{C_2}(\text{client}))$$

```
select C1.*, C2.*
from client as C1, client as C2
where C1.actif = C2.actif
and C1.cID < C2.cID;
```

Calcul des relations

- quelques lois de cette algèbre

$$R \cap S = R - (R - S)$$

$$R \bowtie_C S = \sigma_C(R \times S)$$

$$R \bowtie S = \pi_L(\sigma_C(R \times S))$$

où $C \equiv R.A_1 = S.A_1 \ \wedge \ R.A_2 = S.A_2 \ldots R.A_n = S.A_n$

et $A_1, A_2, \ldots A_n$ sont les attributs communs à R et S

et L est la suite des attributs de R suivi de l'ensemble des attributs de S ne figurant pas dans R

Conception d'une base de données

- on peut utiliser des outils de haut niveau (UML, objets, etc) et générer un calcul de relations à partir de ces formalismes
- on peut directement concevoir des relations… lesquelles ?
- exemple: candidature d'étudiants à une université

ssID nom (numéro sécu, nom)

univ (nom de l'université)

lycée villeL (nom et ville du lycée de provenance)

hobby (plusieurs possibles)

On peut donc construire l'unique relation suivante.

candidat (ssID, nom, univ, lycée, villeL, hobbies)

Conception d'une base de données

- supposons que Jean-Jacques veut candidater aux 3 universités (ENS, Polytechnique, Paris Cité). Il a fréquenté 2 lycées Henri Poincaré à Nancy et Louis-le-grand à Paris. Il fait de la natation et aide la Croix Rouge.

- il va apparaître 12 fois dans la relation **candidat** !

  103, JJ, ENS, HP, Nancy, natation
  103, JJ, ENS, HP, Nancy, croixR
  103, JJ, ENS, LLG, Paris, natation
  103, JJ, ENS, LLG, Paris, croixR
  103, JJ, X, HP, Nancy, natation
  103, JJ, …

- plein de redondances
- modifications compliquées
- suppressions inopinées

candidat

ssID	nom	univ	lycee	villeL	hobby
10731	Jean-Jacques	ENS	HP	Nancy	natation
10731	Jean-Jacques	ENS	HP	Nancy	croixRouge
10731	Jean-Jacques	ENS	LLG	Paris	natation
10731	Jean-Jacques	ENS	LLG	Paris	croixRouge
10731	Jean-Jacques	X	HP	Nancy	natation
10731	Jean-Jacques	X	HP	Nancy	croixRouge
10731	Jean-Jacques	X	LLG	Paris	natation
10731	Jean-Jacques	X	LLG	Paris	croixRouge
10731	Jean-Jacques	P7	HP	Nancy	natation
10731	Jean-Jacques	P7	HP	Nancy	croixRouge
10731	Jean-Jacques	P7	LLG	Paris	natation
10731	Jean-Jacques	P7	LLG	Paris	croixRouge
10899	Iteki	ENS	koko1	Tokyo	violon
10899	Iteki	ENS	koko1	Tokyo	badmington
10899	Iteki	P7	koko1	Tokyo	violon
10899	Iteki	P7	koko1	Tokyo	badmington

Conception d'une base de données

- supposons que Jean-Jacques veut candidater aux 3 universités (ENS, Polytechnique, Paris Cité). Il a fréquenté 2 lycées Henri Poincaré à Nancy et Louis-le-grand à Paris. Il fait de la natation et aide la Croix Rouge.

- suppression des redondances

étudiant (ssID, nom)
candidat (ssID, univ)
provenance (ssID, lycée)
adresse (lycée, villeL)
hobbies (ssID, hobby)

étudiant (ssID, nom)
candidat (ssID, univ, hobby)
provenance (ssID, lycée, villeL)

- suppression de relations inutiles
- rajouter un choix du hobby en fonction de la candidature

ssID	hobby
10731	natation
10731	croixRouge
10899	violon
10899	badmington

ssID	lycee
10731	HP
10731	LLG
10899	koko1

lycee	villeL
HP	Nancy
LLG	Paris
koko1	Tokyo

ssID	univ
10731	ENS
10731	X
10731	P7
10899	ENS
10899	P7

ssID	nom
10899	Iteki
10731	Jean-Jacques

ssID	univ	hobby
10731	ENS	croixRouge
10731	X	natation
10731	X	croixRouge
10731	P7	natation
10899	ENS	natation
10899	P7	natation
10899	P7	croixRouge

ssID	lycee	villeL
10731	HP	Nancy
10731	LLG	Paris
10899	koko1	Tokyo

Dépendance fonctionnelle — BCNF

- considérons la relation: candidat (ssID, nom, univ)
- il y a redondance, car ssID fixe le nom
- **dépendance fonctionnelle** entre ssID et le nom: ssID $\mapsto$ nom
- ssID doit être une clé (*key*) de la relation
- faire 2 relations avec ssID comme clé: étudiant (ssID, nom) candidat (ssID, univ)
- forme normale de Boyce-Codd (BCNF)

$A \mapsto B \implies A$ est une clé

Dépendance multiple — *Fourth normal form*

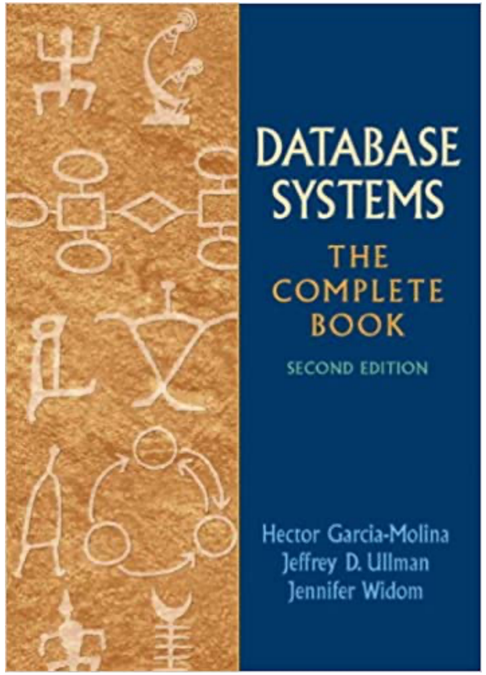
- considérons la relation: candidat (ssID, univ, lycée)
- il n'y a pas de dépendance fonctionnelle
- ssID fixe la combinaison univ et lycée
- **dépendance multiple** entre ssID et univ $\times$ lycée $\quad$ ssID $\mapsto\!\!\!\!\rightarrow$ univ $\quad$ ssID $\mapsto\!\!\!\!\rightarrow$ lycée
- faire 2 relations avec ssID comme clé: $\quad$ candidat (ssID, univ) $\quad$ provenance (ssID, lycée)
- *Fourth normal form*

$$A \mapsto\!\!\!\!\rightarrow B \implies A \text{ est une clé}$$

- 4NF implique BCNF (pas réciproquement)

Toujours plus

- vues et triggers (suite)
- droits d'accès
- au-delà de SQL: graphes, orientation objet, XML



Database Systems
(The Complete Book — 2nd edition)

Hector Garcia-Molina
Jeffrey D. Ullman
Jennifer Widom

Department of Computer Science
Stanford University

Prentice Hall - 2009
ISBN 0-13-606701-8,
978-0-13-606701-6

- le cours de Jennifer Widom

http://cs.stanford.edu/people/widom/DB-mooc.html